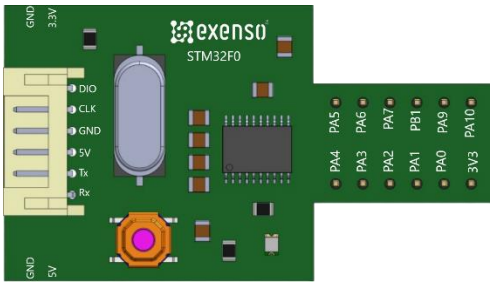
Getting started with an STM32F0 development board using STM32CubeIDE

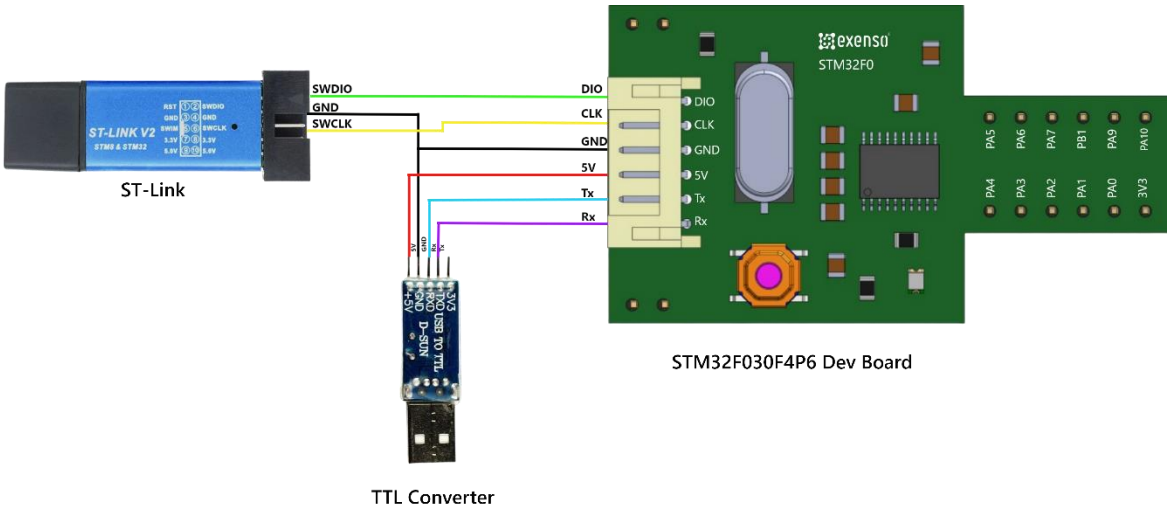
Getting started: Choose a development board STM32F030F4P6



Features of STM32F030F4P6

- 32-bit ARM® Cortex®-M0 CPU
- Maximum 48MHz system main frequency
- 4KB SRAM with hardware parity check
- 16KB Flash
- Operating voltage: 2.4V - 3.6V
- Internal 8MHz RC oscillator (HSI) with PLL option
- Internal 40kHz RC oscillator (LSI)
- External 4-32MHz crystal oscillator (HSE)
- Multiple low-power modes: Sleep, Stop, Standby
- Power on/down reset
- 15 I/Os, all available as external interrupts, many 5V-tolerant
- 1x 5-channel DMA controller with flexible mapping
- 1x 12-bit ADC, up to 9+2 channels, 0V - 3.6V
- 1× 16-bit advanced-control timer with 6 PWM-channels
- 4× 16-bit general-purpose timer
- 2x watchdog timer
- 1x SysTick timer
- 1x real-time clock (RTC) with calendar, alarm, and periodic wakeup
- 1x USART interface with auto baud rate detection
- 1x I^2C interface supporting Fast-mode Plus (1Mbit/s)
- 1x SPI interface with up to 18Mbit/s
- Hardware CRC-32 calculation unit
- Serial wire debug interface (SWD)
- Factory built-in embedded UART bootloader

Hardware connection with ST-link and TTL



The ST-LINK acts as an in-circuit debugger and programmer. This means it allows you to transfer your compiled code from your computer to the microcontroller's internal memory. Without a programmer like the ST-LINK, you wouldn't be able to get your program onto the microcontroller to run it.

TTL (Transistor–transistor logic) converter is used for UART communication. UART (Universal Asynchronous Receiver Transmitter) is a serial communication protocol commonly found in microcontrollers and embedded systems. It uses TTL logic levels for data transmission. And power up our STM32F0 Dev board.

Using STM32CubeIDE

Create STM32 new project

Click STM32Cube IDE Desktop Icon Figure 1

Figure 1: Open Cube IDE

Select or Browse a file path that will create the stm32 project files. Then Click Launch

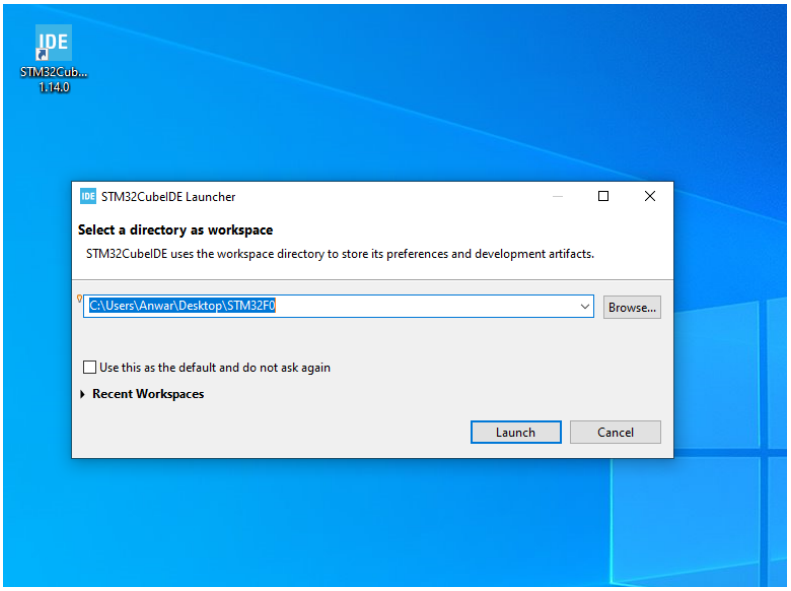


Figure 2: Select a file path as a workspace

The first time opening STM32CubeIDE, it will show an initial screen that has links to guides and other resources that help you get started.

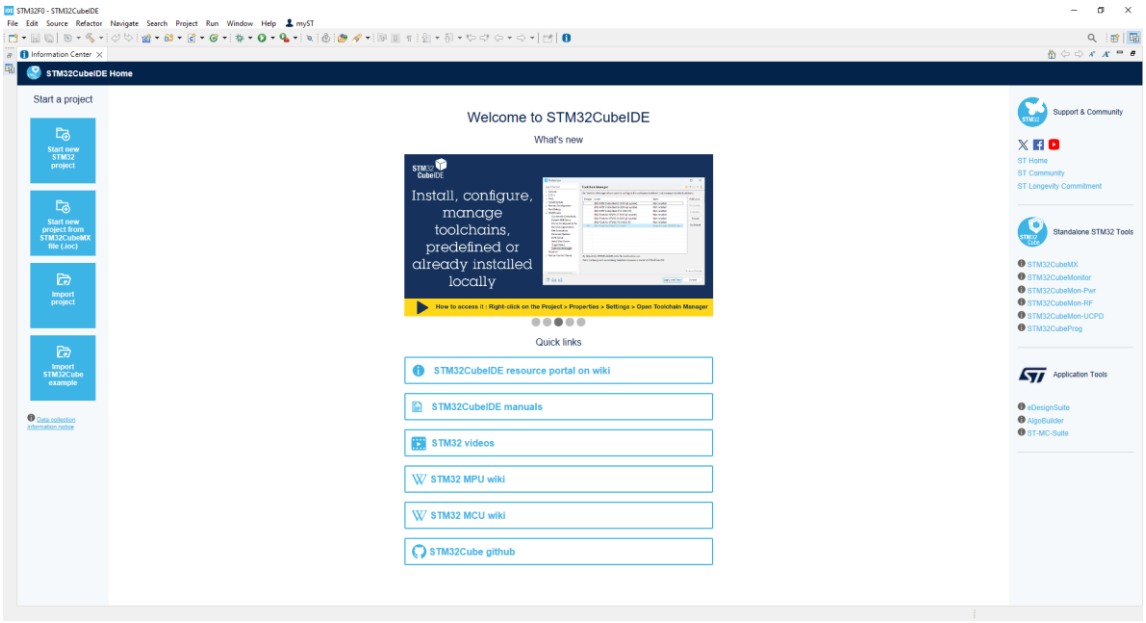


Figure 3: Welcome Screen

Click **"Start new STM32 Project"** and cube Ide download Necessary Driver files Figure 4

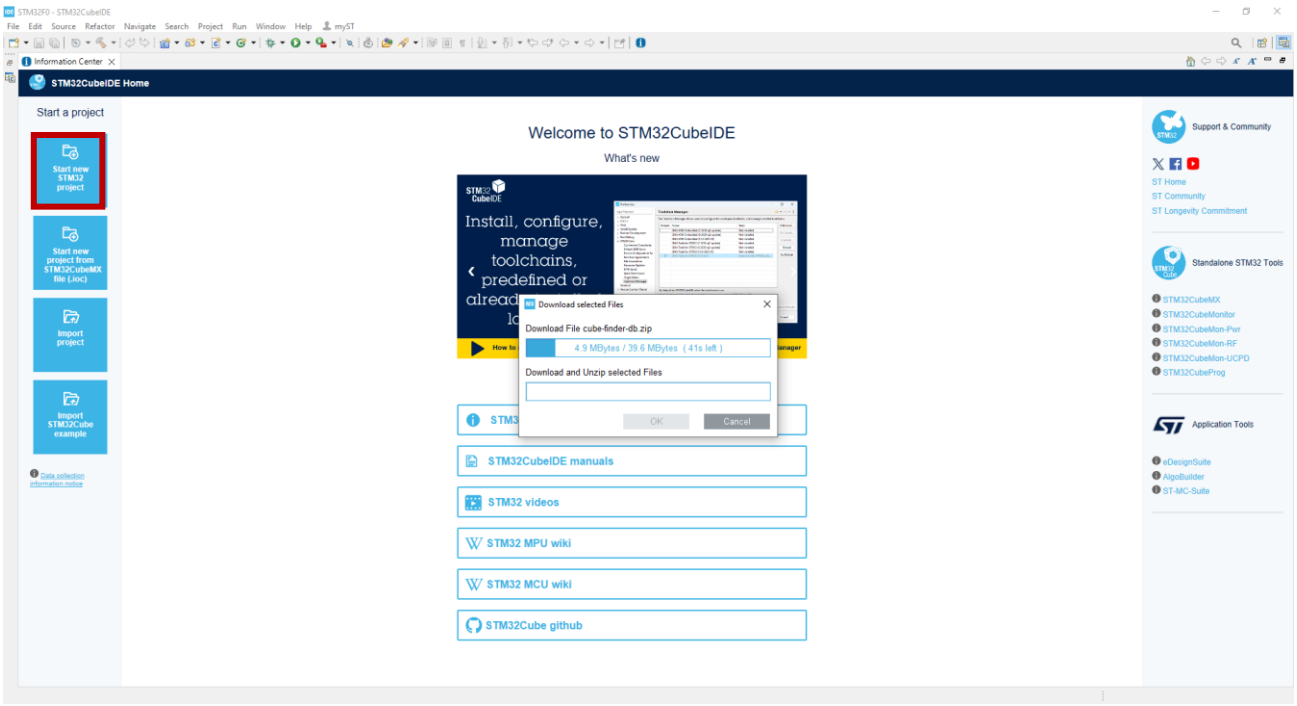


Figure 4: Download Files

When Files Download Completed shows a pop-up window “Target Selection” Figure 5

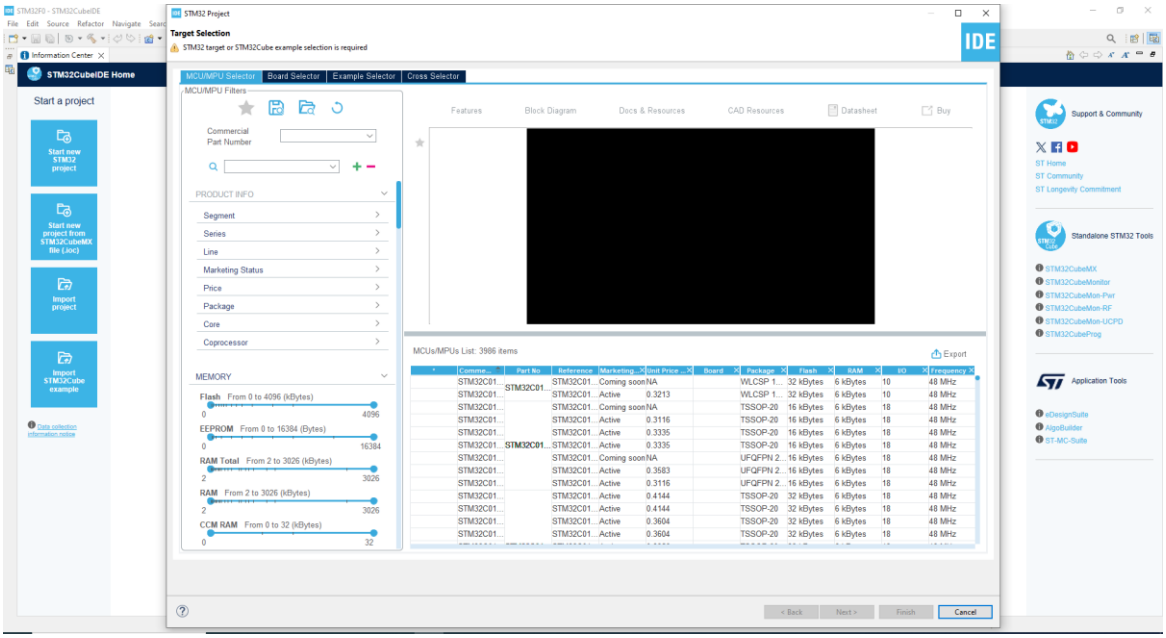


Figure 5: Target Selection

Target Selection Window Click the commercial Part Number and Type “STM32F030F4P6” Figure 6

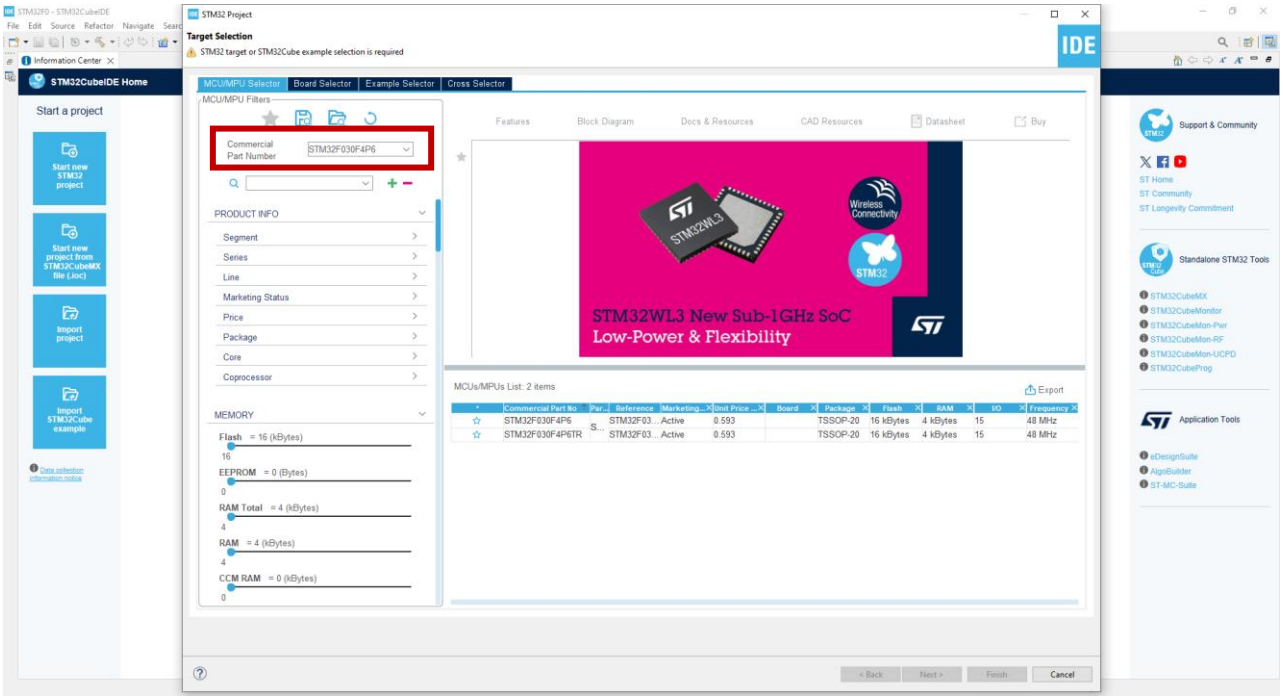


Figure 6: Commercial Part Number

Then Click MCUs/MPC List select First List and Tap “Next” Figure 7

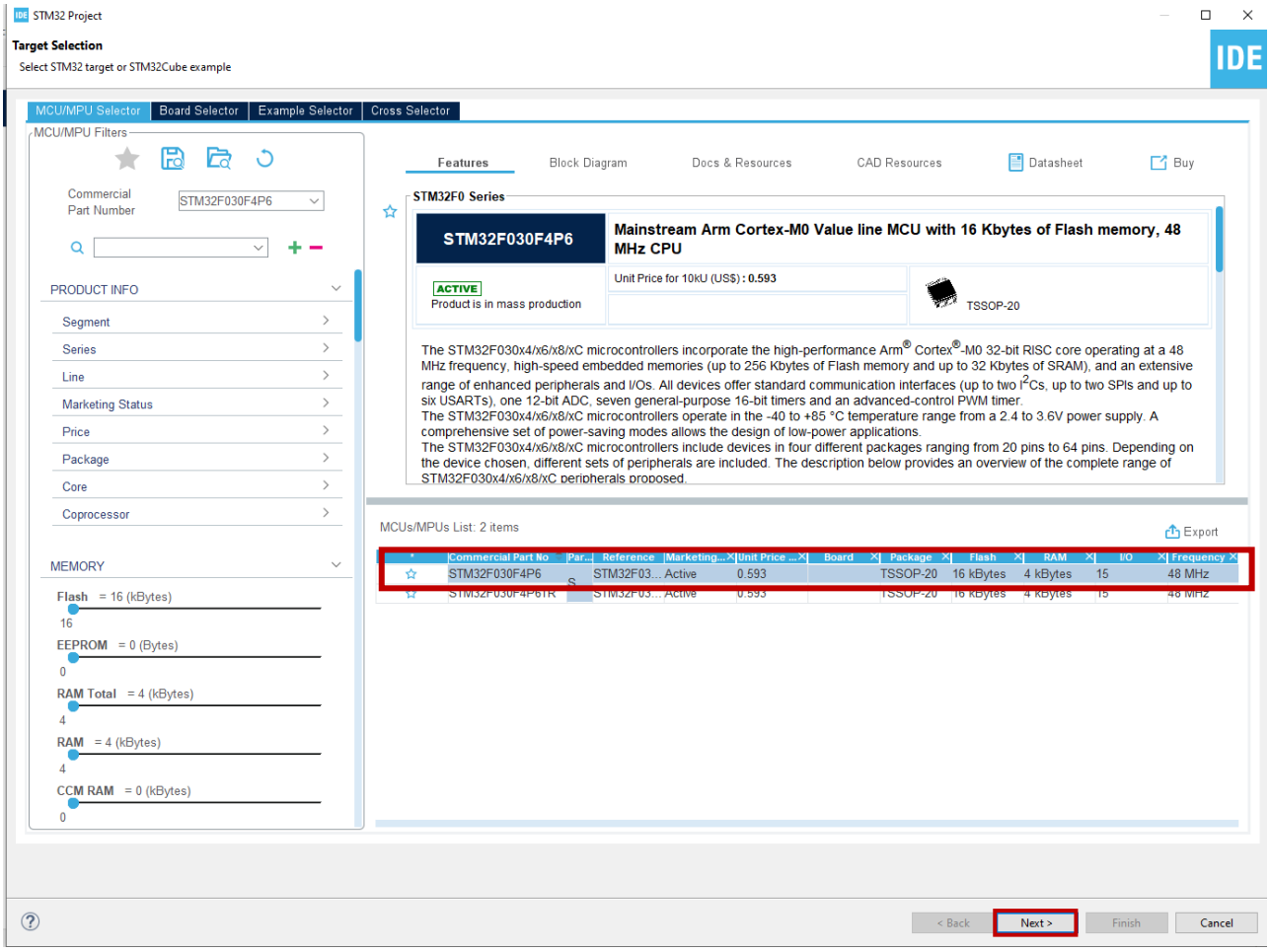


Figure 7: MCUs/MPU List

When you press next then shows a pop-up window creates a valid project name and click **“Finish”** Figure 8

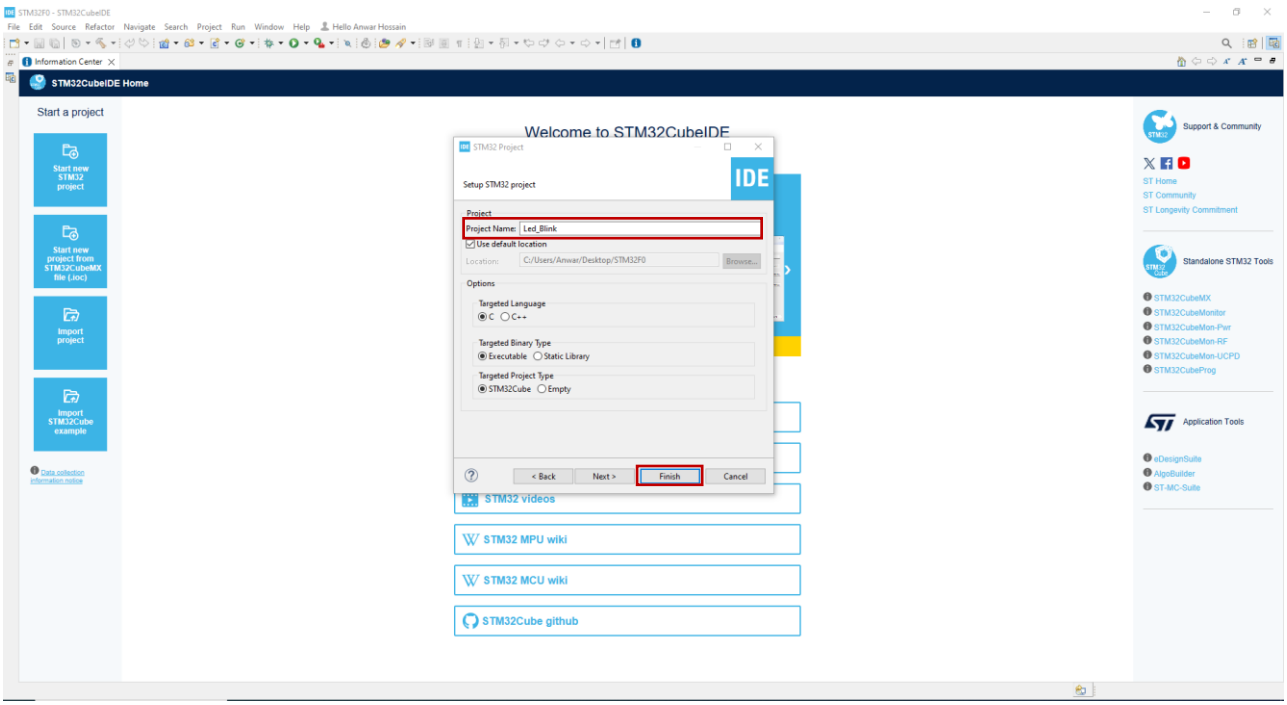


Figure 8: Create Project Name

Then Firmware Library Package Setup shows the pop-up window, click **“Yes”** Figure 9

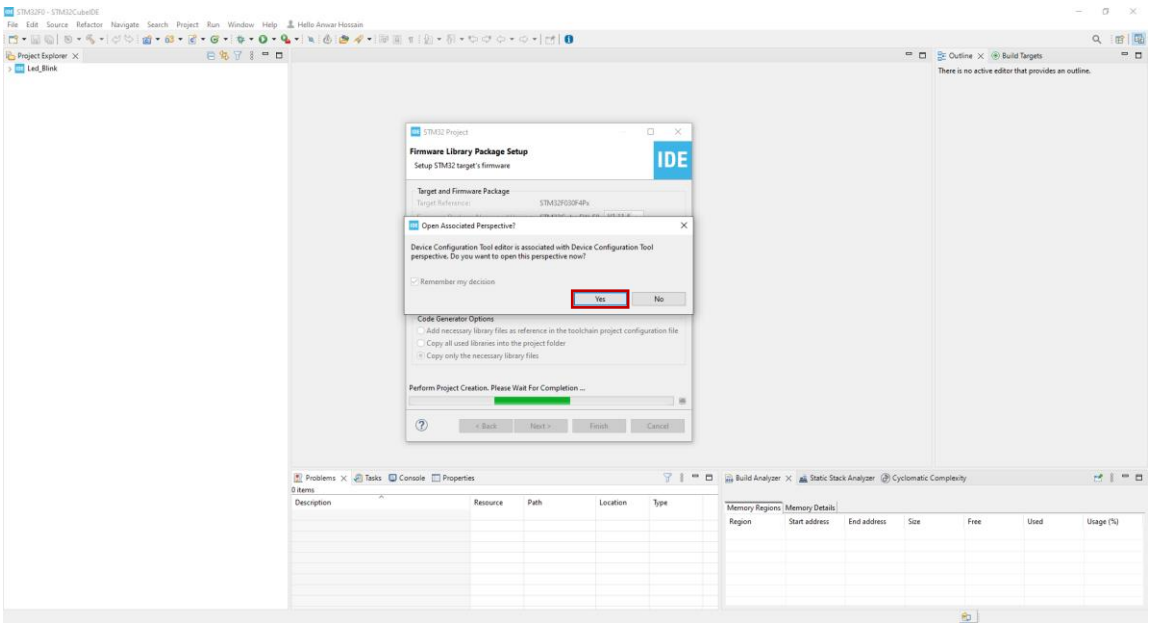


Figure 9: Create Project Name

Appearing Led_Blink.ioc (your_project_name.ioc) - Pinout & Configuration screen. Figure 10

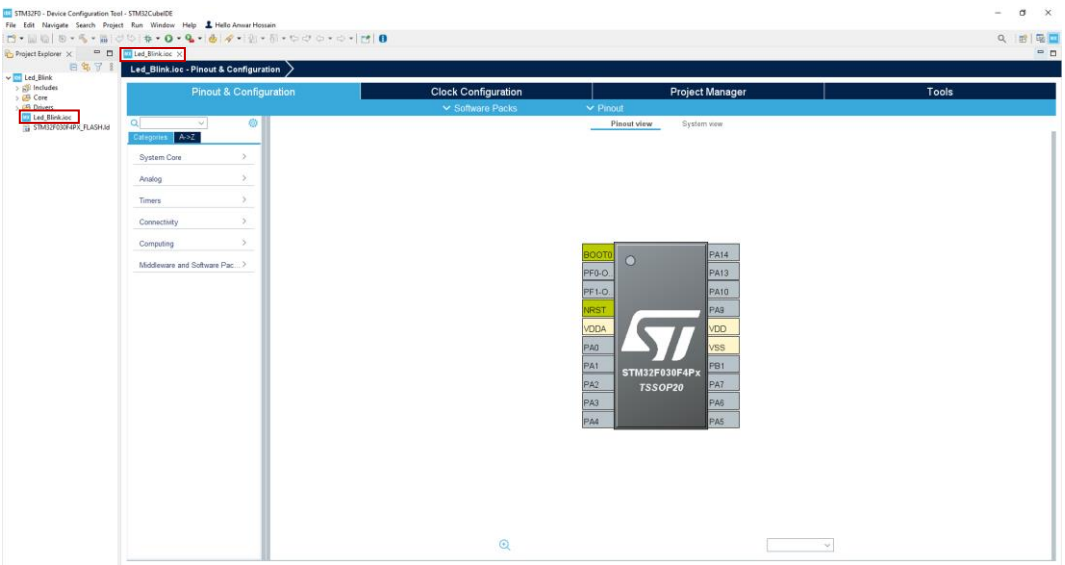


Figure 10: Pinout & Configuration window

Press “**System Core”** then show the drop-down list and select **“SYS”** Figure 11

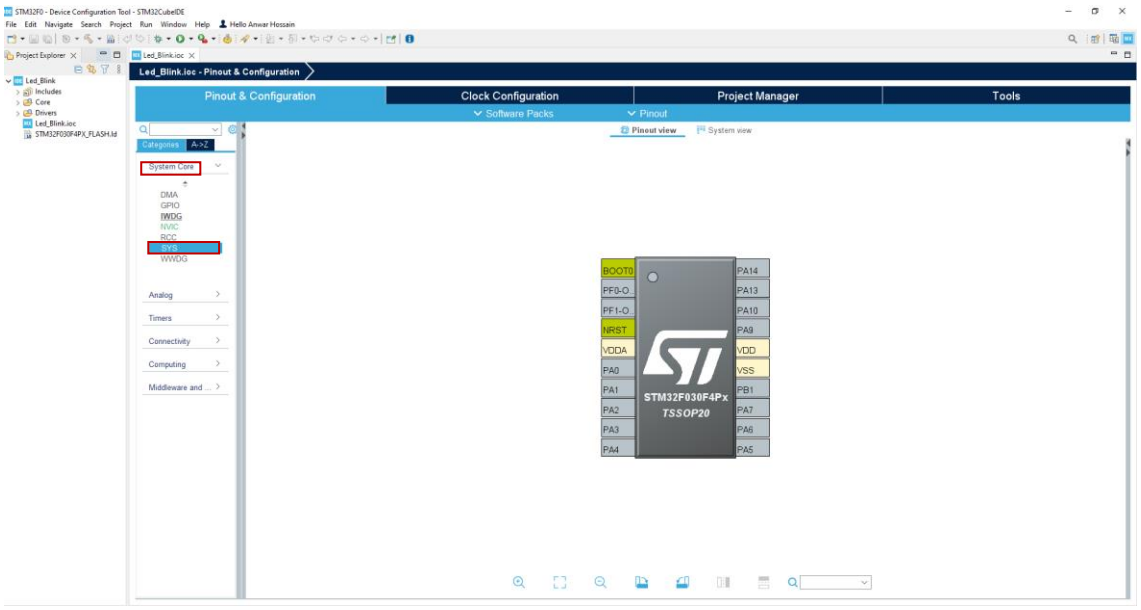


Figure 11: System Core

Mark **“Debug Serial Wire”** Figure 12

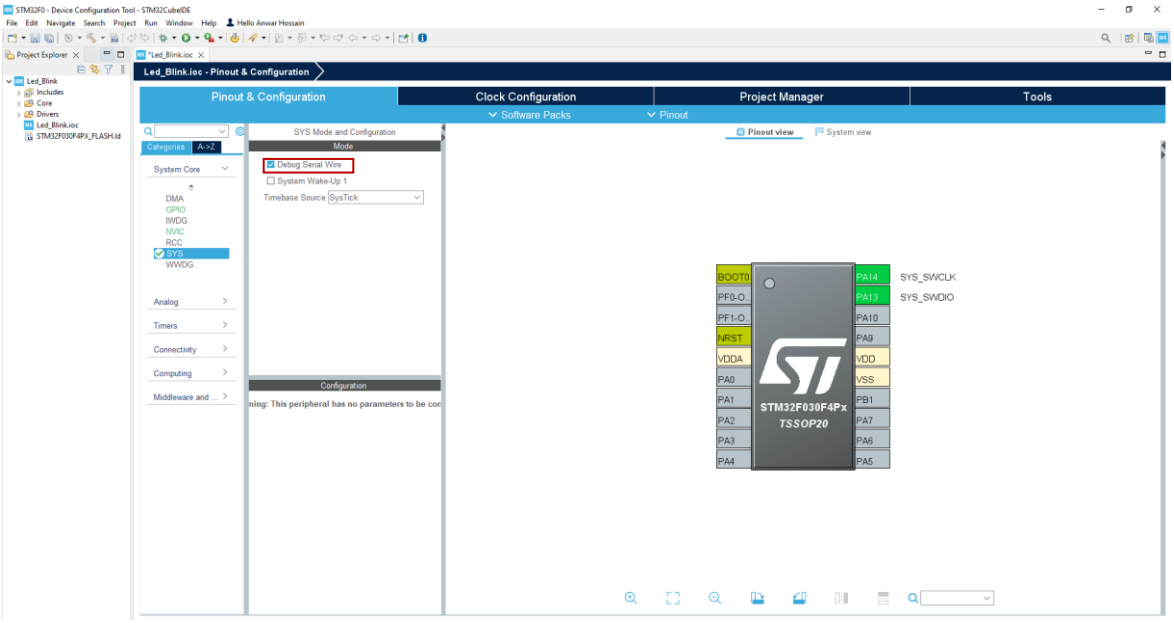


Figure 12: SYS Mode and Configuration

Click **“Clock Configuration”** Then shows Figure 13

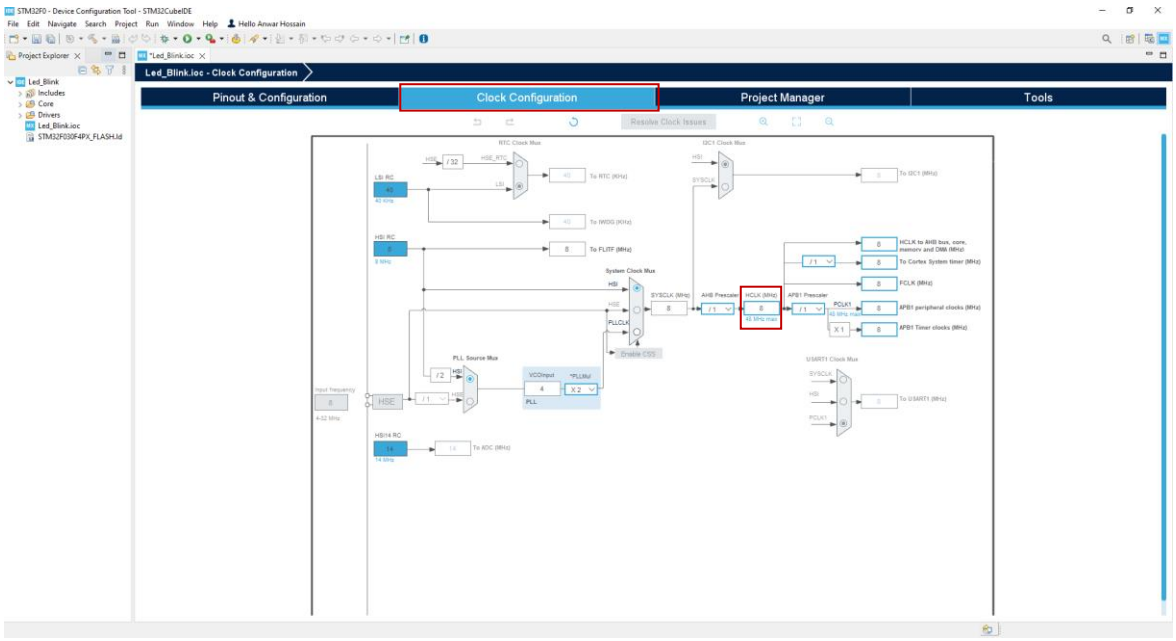


Figure 13: Clock Configuration

Change **HCLK** value from 8 MHz to 48 MHz (max) value and tap **“OK”** Figure 14

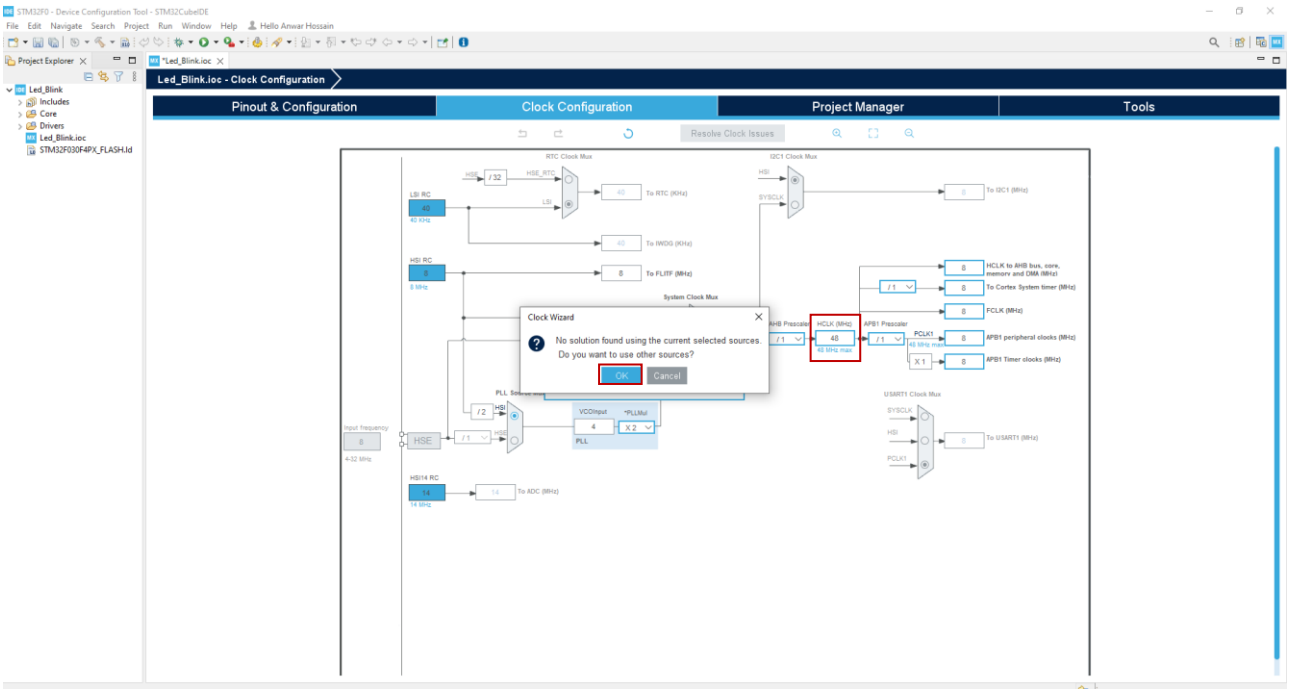
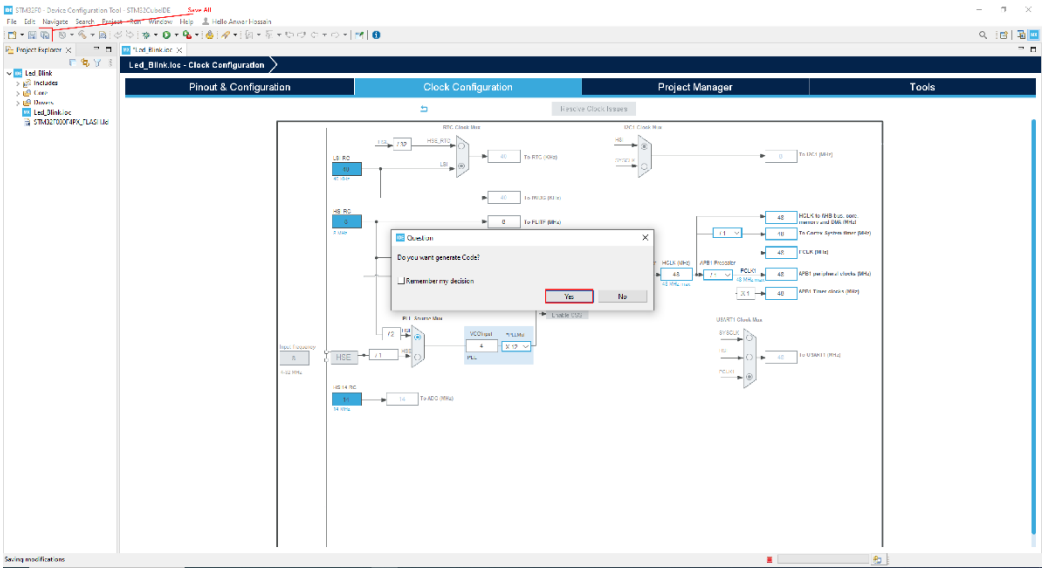


Figure 14: Change Clock Frequency

Click **“Save All”** then open the pop-up window show **“Do you want generate Code?”** tap **“Yes”** then next pop-up tap **“Yes”** Figure 15



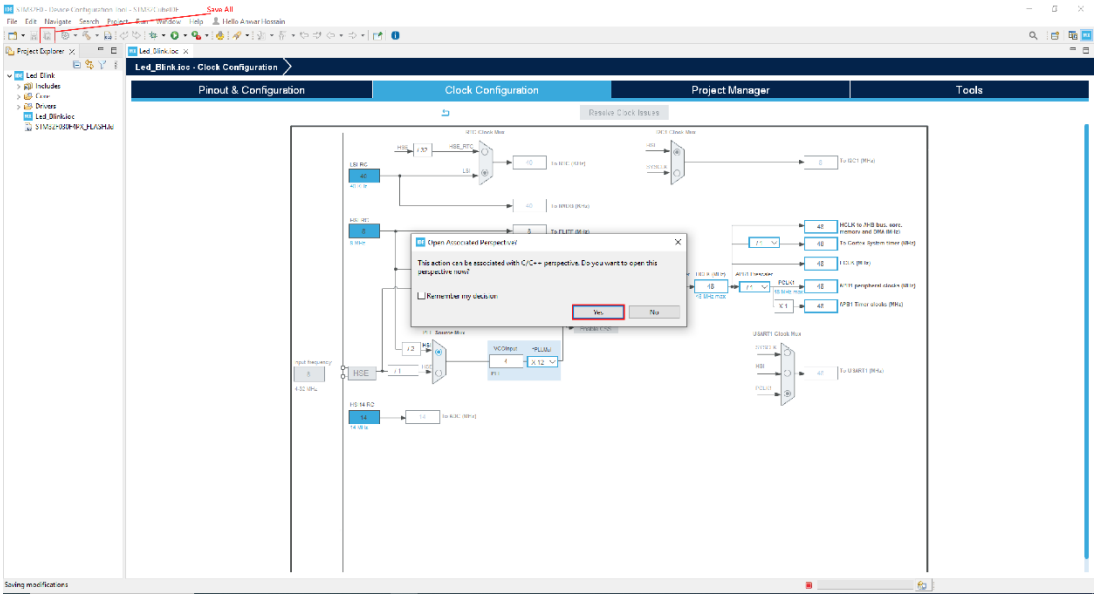
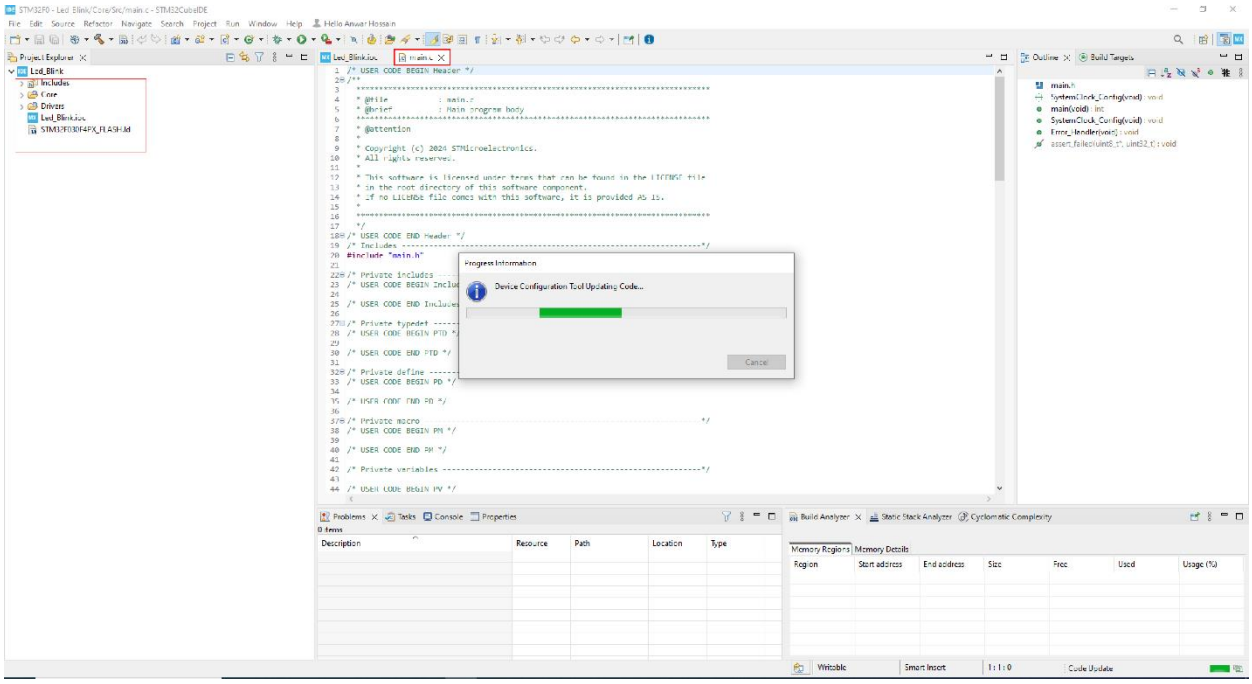


Figure 15: Code Generate

Finally shows the main.c file and other Lib, Driver files generated



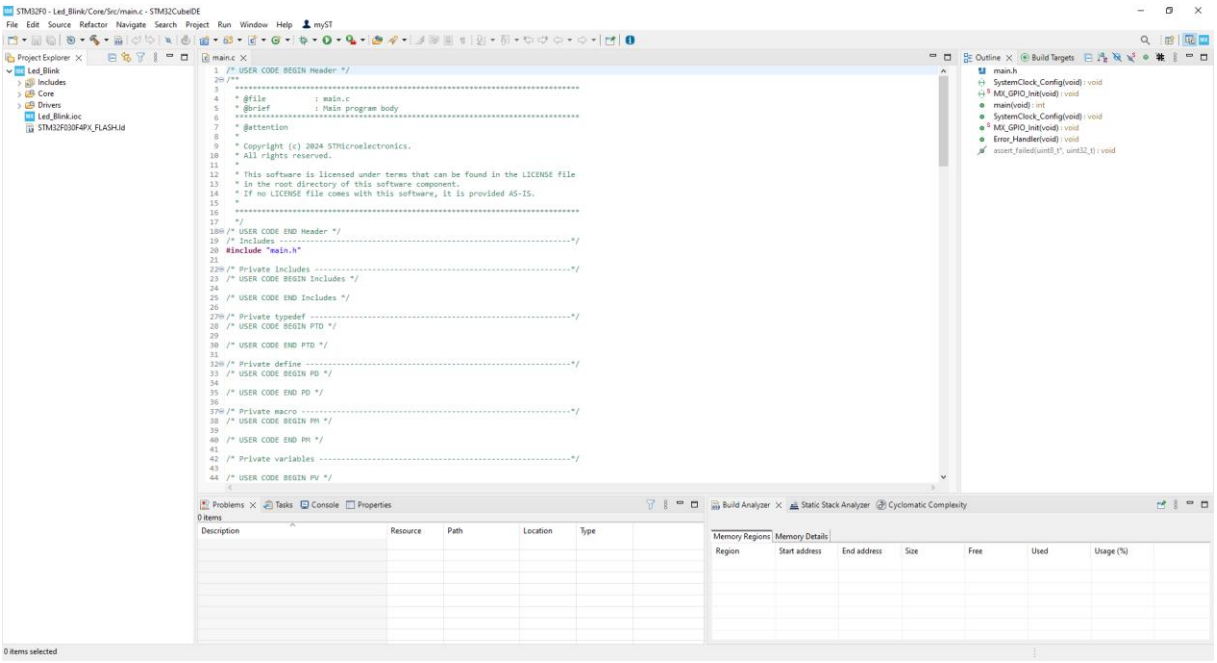
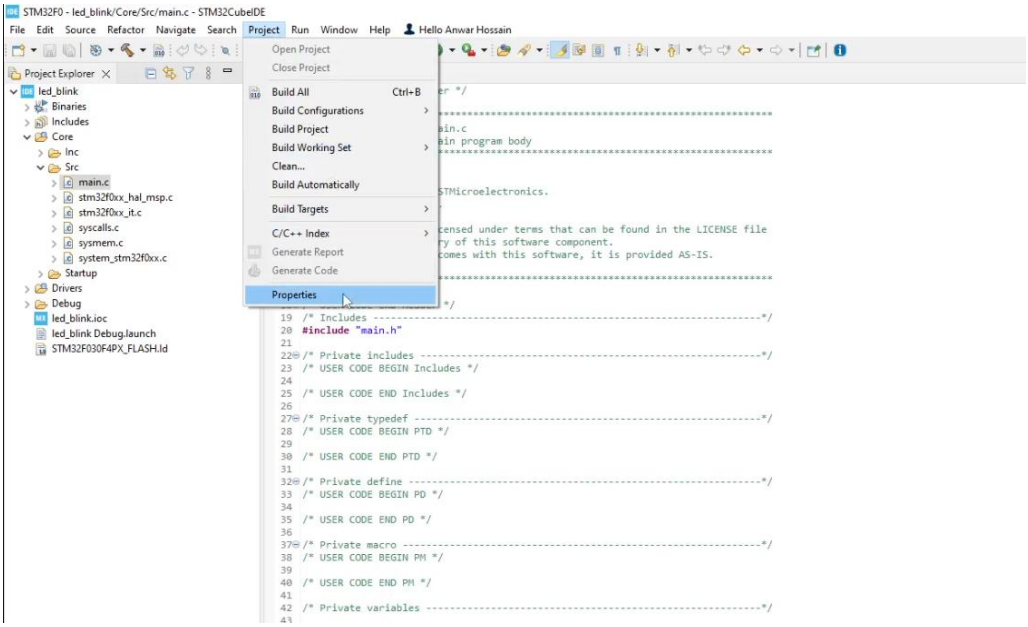
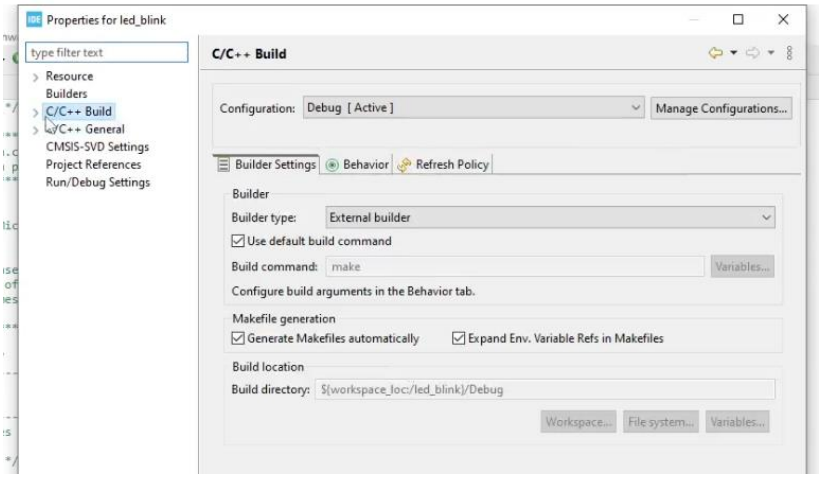
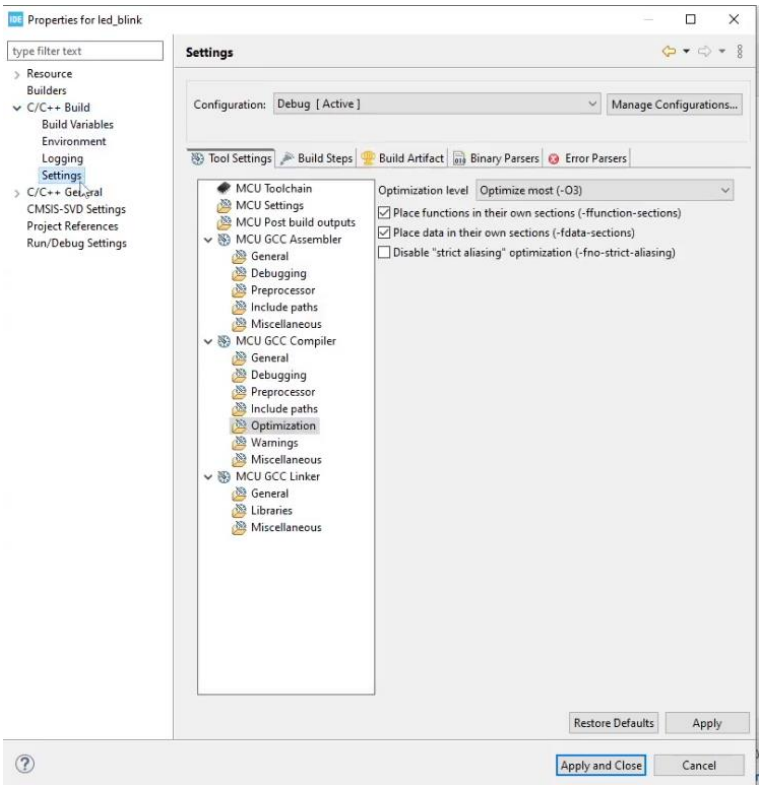


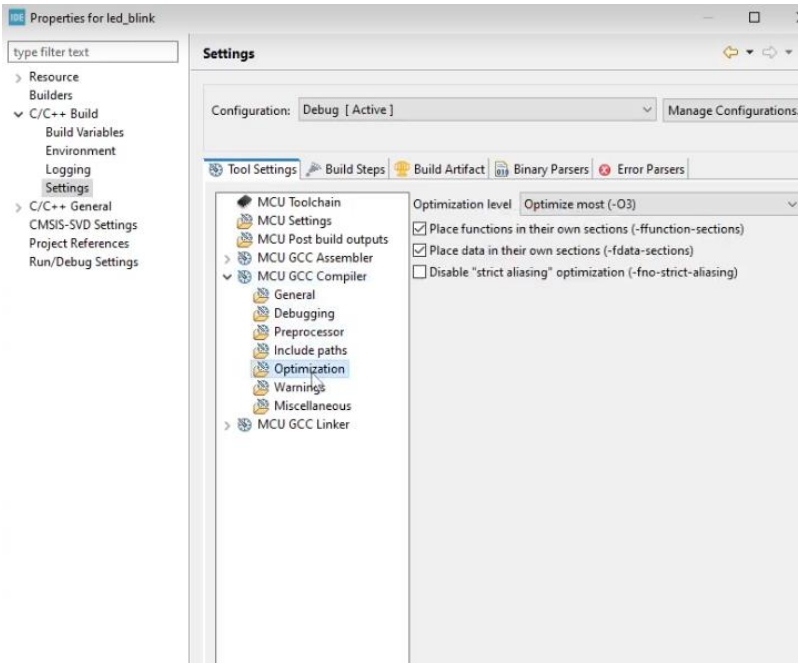
Figure 16: Open main.c file window

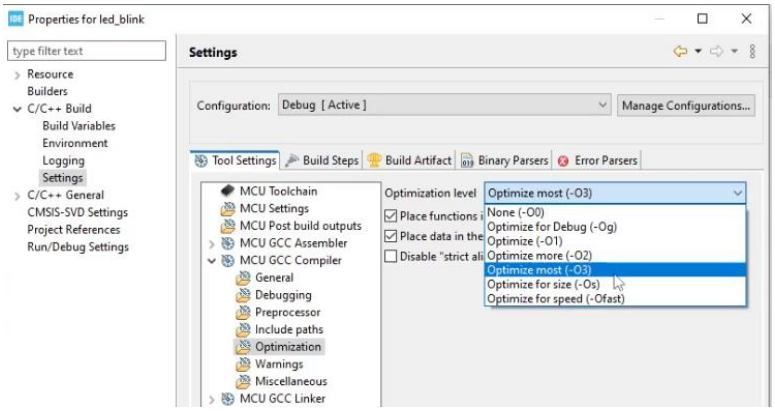
Code Optimization

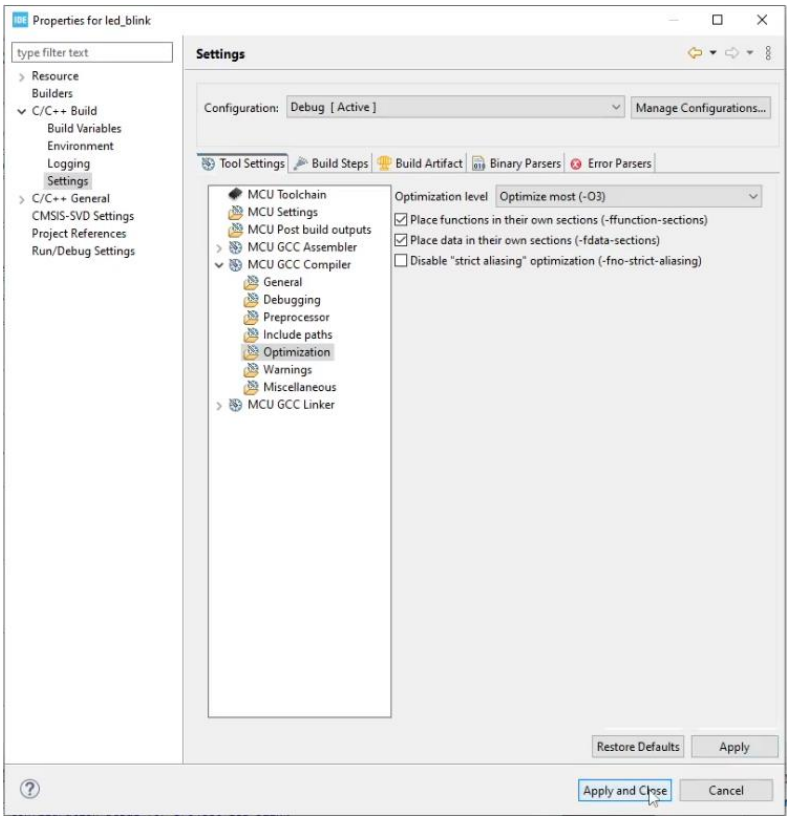







Properties for led_blink
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
C/C++ General
CMSIS-SVD Settings
Project References
Run/Debug Settings
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Build Steps
Build Artifact
Binary Parsers
Error Parsers
MCU Toolchain
MCU Settings
MCU Post build outputs
MCU GCC Assembler
MCU GCC Compiler
General
Debugging
Preprocessor
Include paths
Optimization
Warnings
Miscellaneous
MCU GCC Linker
Optimization level
Optimize most (-O3)
Place functions in their own sections (-ffunction-sections)
Place data in their own sections (-fdata-sections)
Disable "strict aliasing" optimization (-fno-strict-aliasing)


Properties for led_blink
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
C/C++ General
CMSIS-SVD Settings
Project References
Run/Debug Settings
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Build Steps
Build Artifact
Binary Parsers
Error Parsers
MCU Toolchain
MCU Settings
MCU Post build outputs
MCU GCC Assembler
MCU GCC Compiler
General
Debugging
Preprocessor
Include paths
Optimization
Warnings
Miscellaneous
MCU GCC Linker
Optimization level
Optimize most (-O3)
None (-O0)
Optimize for Debug (-Og)
Optimize (-O1)
Optimize more (-O2)
Optimize most (-O3)
Optimize for size (-Os)
Optimize for speed (-Ofast)


Properties for led_blink
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
C/C++ General
CMSIS-SVD Settings
Project References
Run/Debug Settings
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Build Steps
Build Artifact
Binary Parsers
Error Parsers
MCU Toolchain
MCU Settings
MCU Post build outputs
MCU GCC Assembler
MCU GCC Compiler
General
Debugging
Preprocessor
Include paths
Optimization
Warnings
Miscellaneous
MCU GCC Linker
Optimization level
Optimize most (-O3)
Place functions in their own sections (-ffunction-sections)
Place data in their own sections (-fdata-sections)
Disable "strict aliasing" optimization (-fno-strict-aliasing)
Restore Defaults
Apply
Apply and Close
Cancel